Patent Application
Docket No. 002.0259.01
NAI Docket No. 01.162.01

SYSTEM AND METHOD FOR PROVIDING PASSIVE SCREENING OF TRANSIENT MESSAGES IN A DISTRIBUTED COMPUTING ENVIRONMENT

Cross-Reference to Related Applications

This patent application is a conversion of U.S. provisional patent applications, Serial No. 60/309,835, filed August 3, 2001, pending; and Serial No. 60/309,858, filed August 3, 2001, pending; the priority dates of which are claimed and the disclosures of which are incorporated by reference.

Field of the Invention

The present invention relates in general to passive message screening and, in particular, to a system and method for providing passive screening of transient messages in a distributed computing environment.

Background of the Invention

Computer viruses, or simply "viruses," are executable programs or procedures, often masquerading as legitimate files, messages or attachments that cause malicious and sometimes destructive results. More precisely, computer viruses include any form of self-replicating computer code which can be stored, disseminated, and directly or indirectly executed by unsuspecting clients. Viruses travel between machines over network connections or via infected media and can be executable code disguised as application programs, functions, macros, electronic mail (email) attachments, images, applets, and even hypertext links.

The earliest computer viruses infected boot sectors and files. Over time, computer viruses became increasingly sophisticated and diversified into various genre, including cavity, cluster, companion, direct action, encrypting, multipartite, mutating, polymorphic, overwriting, self-garbling, and stealth viruses, such as described in "Virus Information Library," http://vil.mcafee.com/default.asp?, Networks Associates Technology, Inc., (2001), the disclosure of which is incorporated by reference. Macro viruses are presently the most popular form of virus. These viruses are written as scripts in macro programming languages, which are often included with email as innocuous-looking attachments.

The problems presented by computer viruses, malware, and other forms of bad content are multiplied within a bounded network domain interfacing to external internetworks through a limited-bandwidth service portal, such as a gateway, bridge or similar routing device. The routing device logically forms a protected enclave within which clients and servers exchange data, including email and other content. All data originating from or being sent to systems outside the network domain must pass through the routing device. Maintaining high throughput at the routing device is paramount to optimal network performance.

Routing devices provide an efficient solution to interfacing an intranetwork of clients and servers to external internetworks. Most routing devices operate as store-and-forward packet routing devices, which can process a high volume of traffic transiting across the network domain boundary. Duplicate messages, however, introduce inefficiencies and can potentially degrade performance. For example, a message can be sent with multiple recipients who each receive a separate copy. Nevertheless, the routing device must process each duplicate message as if the message were unique.

A firewall can be used with a routing device to provide limited security. The firewall filters incoming packets to deny access by unauthorized users. Thus, the firewall can protect indirectly against the introduction of computer viruses and other malware into a network domain. As each duplicate message must still be scanned prior to delivery, a firewall does not relieve packet congestion at a network boundary and can actually degrade throughput by delaying delivery.

The bottleneck created by the routing device and firewall create a security risk that can be exploited in a denial of service (DoS) attack. The "ILOVEYOU" virus, released in May 2000, dramatically demonstrated the vulnerability of network infrastructure components by propagating copies of emails containing the virus using addresses obtained from a user address book on each client system. Each email message contained identical content but listed a different recipient. The resultant email flood saturated servers with massively duplicated copies of substantially the same email and denied service through resource depletion and network bandwidth consumption.

Most firewalls failed to detect the presence of the "ILOVEYOU" virus. Firewalls require *a priori* knowledge of network addresses corresponding to proscribed servers to effectively filter out potentially bad packets. Therefore, infected emails were delivered and unwittingly opened by unsuspecting users, creating a flood of infected message traffic.

Active packet scanners can be used in lieu of or in addition to firewalls to dynamically analyze an incoming packet stream at a network domain boundary. Each packet is intercepted and analyzed while in transit into a protected enclave. However, active scanners can adversely affect the timing of packet delivery. Detecting computer viruses, malware and other bad content embedded in upper layer network protocols, in particular, at the transport and application layers, requires the analysis of a stream of lower layer packets collectively comprising upper layer messages. Interrupting the flow of the packet stream, though, can cause the recipient client to timeout and erroneously generate a retransmission request, thereby hindering throughput. As well, active scanners are installed at the gateway and require administrator-level permissions and privileges and create further potential security risks.

Therefore, there is a need for an approach to passively screening a multiplicity of substantially duplicate message packets transiting the boundary of a network domain. Preferably, such an approach would detect protocol-specific computer viruses, malware and other bad content without causing an interruption of an incoming data packet stream.

There is a further need for an approach to identifying patterns in a transient packet stream for events indicative of a network service or similar type of attack, preferably in combination with the detection of computer viruses, malware and other bad content.

Summary of the Invention

The present invention provides a system and method for passively detecting computer viruses, malware, and bad content in transient packets and denial of service and related network attacks. Incoming network packets are passively copied from a packet stream into an incoming message queue. The network packets in the stream are reassembled into and identified as upper layer network protocol packets. Each reassembled packet is scanned by a network protocol-specific scanner to identify computer viruses, malware, and other bad content. Concurrently, the reassembled packets are analyzed as a packet stream to identify a denial of service or related type of network attack. By passively processing the incoming packet stream, each packet can be examined for potential security without affecting network throughput.

An embodiment provides a system and a method for providing passive screening of transient messages in a distributed computing environment. A transient packet stream is passively monitored at a network boundary. Incoming datagrams structured in compliance with a network protocol layer are received. One or more of the incoming datagrams are reassembled into a segment structured in compliance with a transport protocol layer. Contents of the reassembled segment are scanned for a presence of at least one of a computer virus and malware to identify infected message contents.

A further embodiment provides a system and method for passively detecting computer viruses and malware and denial of service-type network attacks in a distributed computing environment. Copies of datagrams transiting a boundary of a network domain are received into an incoming packet queue. Each datagram is copied from a packet stream. One or more such datagrams from the incoming packet queue are reassembled into network protocol packets. Each network protocol packet is staged in a reassembled packet queue. Each network protocol packet from the reassembled packet queue is scanned to ascertain an infection of at least one of a computer virus and malware. Events identified from the datagrams in the packet stream are evaluated to detect a denial of service-type network attack on the network domain.

Still other embodiments of the present invention will become readily apparent to those skilled in the art from the following detailed description, wherein is described embodiments of the invention by way of illustrating the best mode contemplated for carrying out the invention. As will be realized, the invention is capable of other and different embodiments and its several details are capable of modifications in various obvious respects, all without departing from the spirit and the scope of the present invention. Accordingly, the drawings and detailed description are to be regarded as illustrative in nature and not as restrictive.

Brief Description of the Drawings

FIGURE 1 is a block diagram showing a system for providing passive screening of transient messages in a distributed computing environment, in accordance with the present invention.

FIGURE 2 is a functional block diagram showing the software modules of the antivirus system of FIGURE 1.

FIGURE 3 is a functional block diagram showing the protocol-specific message queue of the antivirus system of FIGURE 2.

FIGURE 4 is a process flow diagram showing the passive screening of a transient message using the antivirus system of FIGURE 1.

FIGURE 5 is a block diagram showing the interrelationships between transient packets processed by the antivirus system of FIGURE 1.

FIGURE 6 is a flow diagram showing a method for providing passive screening of transient messages in a distributed computing environment, in accordance with the present invention.

FIGURE 7 is a flow diagram showing the routine for receiving a packet for use in the method of FIGURE 6.

FIGURE 8 is a flow diagram showing the routine for reassembling a packet for use in the routine of FIGURE 7.

FIGURE 9 is a flow diagram showing the routine for scanning a message for use in the method of FIGURE 6.

FIGURE 10 is a flow diagram showing the routine for processing an infection for use in the routine of FIGURE 9.

FIGURE 11 is a flow diagram showing the routine for correlating events for use in the method of FIGURE 6.

Detailed Description

FIGURE 1 is a block diagram showing a system for providing passive screening of transient messages in a distributed computing environment 10, in accordance with the present invention. By way of example, a gateway 15 (or bridge, router, or similar packet routing device) interfaces an intranetwork 14 to an internetwork 16, including the Internet. The intranetwork 14 interconnects one or more servers 12 with one or more clients 11a-b within a bounded network domain defined by a common network address space. The server 12 includes a storage device 13 for common file storage and sharing. The clients 11a-b can also include storage devices (not shown).

The individual servers 12 and clients 11a-b externally connect to one or more remote servers 17 and remote clients 19 over the internetwork 16 via the gateway 15. The gateway 15 operates as a store-and-forward packet routing device, which processes a high volume of packet traffic transiting across the network domain boundary. The gateway 15 provides an efficient solution to interfacing the individual servers 12 and clients 11a-b to external systems operating over the internetwork 16. Optionally, a firewall 20 can provide limited security to the intranetwork 14 by providing filtering of packets originating from unauthorized users. Other network topologies and configurations are feasible, as would be recognized by one skilled in the art.

In addition to the firewall 20, an antivirus system (AVS) 21 passively analyzes message packets incoming to the bounded network domain for the presence of computer viruses, malware, and other bad content, and provides passive screening of a transient packet stream, as further described below with reference to FIGURE 2. Each component in the distributed computing environment 10 executes a layered network protocol stack for processing different types of packets, including packets compliant with the Internet Protocol (IP) and Transmission Control Protocol (TCP).

The individual computer systems, including servers 12, 17 and clients 11a-b, 19 are general purpose, programmed digital computing devices consisting of a central processing unit (CPU), random access memory (RAM), non-volatile secondary storage, such as a hard drive or CD ROM drive, network interfaces, and peripheral devices, including user interfacing means, such as a keyboard and display. Program code, including software programs, and data are loaded into the RAM for execution and processing by the CPU and results are generated for display, output, transmittal, or storage.

FIGURE 2 is a functional block diagram showing the software modules 30 of the antivirus system 21 of FIGURE 1. The antivirus system 21 includes three functionally separate modules: event correlator 31, antivirus scanner 32, packet receiver 33, and network interface 34. The network interface 34 operates in a promiscuous mode to copy each incoming packet 43 into an incoming packet queue 42. The transient message packets are preferably exchanged in compliance with the SMTP protocol, such as described in W.R. Stevens, "TCP/IP Illustrated, Vol. 1, The Protocols," Ch. 28, Addison Wesley Longman, Inc. (1994), the disclosure of which is incorporated by reference.

The packet receiver 33, antivirus scanner 32, and event controller 31 are functionally separate modules. The packet receiver 33 retrieves the incoming packets 43 from the incoming packet queue 42. The packet receiver 33 operates at a network protocol layer. In the described embodiment, only packets compliant with the IP protocol are processed. The incoming packets 43 are reassembled by a reassembler submodule 39 into TCP protocol layer segments which are then parsed by a parser 40 to identify the specific upper layer protocol employed. The reassembled packets are staged in protocol-specific queues 41, as further described below with reference to FIGURE 3.

The antivirus scanner 32 includes a plurality of protocol-specific scanning submodules 35-38, including submodules for the Hypertext Protocol (HTTP), File Transfer Protocol (FTP), Simple Mail Transport Protocol (SMTP), and Network

News Transport Protocol (NNTP), although other upper layer network protocols could also be implemented, as would be recognized by one skilled in the art.

Through each protocol-specific submodule 35-38, the antivirus scanner 32 retrieves each re-assembled packet from the appropriate protocol-specific queue 41 for scanning using standard antivirus techniques, as are known in the art. Upon detecting the presence of an infected message, the antivirus scanner 32 logs the occurrence in a log 47. In addition, the antivirus scanner 32 can optionally generate a warning 46 to the network administrator or other appropriate user. As well, the antivirus scanner 32 can optionally "spoof" the origin server by sending a legitimate packet in place of the infected packet. The legitimate packet is placed as an outgoing packet 49 in the outgoing packet queue 48 for sending over the internetwork via the network interface 34.

The antivirus scanner 32 operates in an event-based manner by processing reassembled packets in the appropriate protocol-specific queue 41. The protocol-specific queues 41 function as event-handlers by creating logical connections between the packet receiver 33 and the antivirus scanner 32. Protocol-specific queues 41 provide an intermediate store in which reassembled packets are staged based on the upper layer protocol employed.

The antivirus scanner 32 can fall behind in processing if the protocol-specific queues 41 become saturated with reassembled packets. As the packet receiver 33 can process transient messages at a higher rate than the antivirus scanner 32, the packet receiver 33 maintains the protocol-specific queues 41 at a constant size in pace with the antivirus scanner 32 and prevents protocol-specific queues 41 from becoming saturated by reassembled packets awaiting scanning.

The event correlator 31 optionally provides a meta computer virus screening functionality to the antivirus system 21. The event correlator 31 analyzes the reassembled packets in the protocol-specific queues 41 to identify patterns in the incoming packet stream indicative of a network service attack or other type of network event. Upon detecting an event of interest, the event correlator 31 stores each event 45 in an event database 44.

Each module, including network interface 34, packet receiver 33, antivirus scanner 32, and event correlator 31 is a computer program, procedure or module written as source code in a conventional programming language, such as the C++ programming language, and is presented for execution by the CPU as object or byte code, as is known in the art. The various implementations of the source code and object and byte codes can be held on a computer-readable storage medium or embodied on a transmission medium in a carrier wave. The modules operates in accordance with a sequence of process steps, as further described below with reference to FIGURE 6.

FIGURE 3 is a functional block diagram showing the protocol-specific message queue 40 of the anti-virus system 21 of FIGURE 2. For efficiency, the protocol-specific queue 40 categorizes the individual reassembled packets according to the upper-layer network protocol employed. The anti-virus system 21 supports one protocol-specific queue per upper-layer protocol, although other logical combinations and separations of protocol-specific queues are possible.

Each reassembled packet 51 staged in a protocol-specific queue 40 includes two types of information. First, protocol-dependent information 52 is stored with each reassembled packet 51. The protocol-dependent information 52 includes, by way of example, a source address, source port number, destination address, destination port number, and Uniform Resource Locator (URL) for HTTP; a file name and user name for FTP; and a sender identification, recipient identification, and subject for SMTP. Other types of protocol-dependent information could also be stored, as would be recognized by one skilled in the art.

In addition to the protocol-dependent information 52, the actual content of each reassembled packet is stored as data 53. The data 53 is stored in the data format employed by the specific network protocol. Importantly, while each incoming packet is received as part of a transient packet stream, the only portion of the stream that is actually stored as data 53 is the individual upper layer protocol-packets, incorporating encoding as appropriate. Thus, if ordinarily stored as encoded data, the data 53 would need to be decoded prior to scanning by the antivirus scanner 32. For instance, the SMTP, POP3 and NNTP network protocols require MIME decoding prior to antivirus scanning.

FIGURE 4 is a process flow diagram showing passive screening 60 of a transient message using the antivirus system 21 of FIGURE 1. In the described embodiment, incoming IP datagrams 61 are received from the internetwork. One or more of the incoming IP datagrams 61 are reassembled (step ①) into a reassembled TCP segment 62. The reassembled TCP segment 62 is then parsed. Protocol-dependent information 63 stored (step ②) with the reassembled TCP segment 62. In addition, the actual data 64 is staged (step ③) with the reassembled TCP segment 62. The HTTP, FTP, SMTP, POP3, NNTP, and Gnutella protocol formats are supported, although one skilled in the art would recognize that other protocol formats could also be supported.

FIGURE 5 is a block diagram showing the interrelationships 70 between transient packets processed by the antivirus system 21 of FIGURE 1. The transient packets are structured in compliance with standard TCP/IP network protocol layers, such as described in W. R. Stephens, *TCP/IP Illustrated,* Vol. 1, "The Protocols," Ch. 1-3, cited above.

Raw network data is copied by the antivirus scanner 21 (shown in FIGURE 1) as IP datagrams 74 in a network data layer 71. The IP datagrams 74 are reassembled into TCP segments 75 in a transport protocol layer 72. Finally, the TCP segments 75 are stored as protocol-specific packets, including HTTP packets 76, FTP files 77, SMTP messages 78, NNTP articles 79, and Gnutella files 80, to name a few, in an application protocol layer 73. For simplicity and clarity of presentation, the term *packets* is used to refer generically to files, messages, articles, datagrams, and packets. The described embodiment performs the necessary antivirus scanning on packets of these types through passive packet screening. Thus, the throughput of message traffic through the network domain boundary remains unaffected by on-going antivirus packet analyses.

FIGURE 6 is a flow diagram showing a method for providing passive screening of transient messages in a distributed computing environment 90, in accordance with the present invention. The method initializes and executes three independent processes for receiving packets (block 91), scanning packets (block 92), and correlating events (block 93), as further described below with reference to FIGURES 7, 9 and 11, respectively. Following completion of the foregoing independent processes (blocks 91-93), the method terminates.

FIGURE 7 is a flow diagram showing the routine for receiving a packet 100 for use in the method 90 of FIGURE 6. The purpose of this routine is to receive and parse incoming packets 43 (shown in FIGURE 2) from the incoming packet queue 42.

Thus, the routine 100 begins by initializing internal data structures (block 101). Incoming packets 43 are then iteratively processed (blocks 102-107), as follows. Each incoming packet 43 is received from the incoming packet queue 42 (block 103). The incoming packet 43 is then reassembled (block 104), as further described below with reference to FIGURE 8. In a further embodiment, the incoming packet stream can be stopped if an infected packet is identified. Thus, upon request (block 105), the current incoming packet stream is stopped (block 106). Processing continues with each incoming packet 43 (block 107), after which the routine returns.

FIGURE 8 is a flow diagram showing the routine for reassembling a packet 110 for use in the routine 100 of FIGURE 7. The purpose of this routine is to reassemble incoming IP datagrams 74 into TCP segments 75 (shown in FIGURE 5). The described embodiment specifically reassembles TCP segments from IP datagrams as the TCP network protocol is widely used by applications executing in the network and application protocol layers.

Thus, if the IP datagram 74 does not contain a TCP segment 75 (block 111), the incoming packet 43 is discarded (block 112). Otherwise, if the IP datagram 74 contains the first part of a TCP segment 75 (block 113), a new TCP segment is started (block 114) by staging the first part in temporary storage. If the IP datagram 74 is not the last part of a TCP segment 75 (block 115), the part is added to the current TCP segment (block 116) in temporary storage. Otherwise, if the IP datagram 74 contains the last part of a TCP segment 75 (block 115), the current TCP segment is ended (block 117) and the appropriate upper layer protocol for the TCP segment 75 is determined (block 118). In the described embodiment, the HTTP, FTP, SMTP, POP3, NNTP, and Gnutella upper layer network protocols are supported. The TCP segment 75 is then enqueued into the proper protocol-specific queue 41 (shown in FIGURE 2) (block 119), after which the routine returns.

FIGURE 9 is a flow diagram showing the routine for scanning a message 130 for use in the method 90 of FIGURE 6. The purpose of this routine is to identify protocol-specific indicia of computer viruses and malware in transient upper layer network protocol packets.

Each TCP segment 75 is iteratively processed (blocks 131-139), as follows. First, a TCP segment 75 (shown in FIGURE 5) is retrieved from a protocol-specific queue 41 (shown in FIGURE 2) (block 132). If necessary (block 133), the TCP segment 75 is decoded (block 134). In the described embodiment, the SMTP, POP3, and NNTP application layer network protocols require MIME decoding.

The packet is then scanned for protocol-specific indicia of computer viruses, malware, and other bad content (block 135), as is known in the art. If the packet is infected (block 136), the infection is processed (block 137), as further described below with reference to FIGURE 10. Following scanning and any necessary processing, the TCP segment 75 is discarded (block 138). Processing continues with each remaining TCP segment 75 (block 139), after which the routine returns.

FIGURE 10 is a flow diagram showing the routine for processing an infection 140 for use in the routine 130 of FIGURE 9. The purpose of this routine is to perform one or more actions following the detection of a computer virus infection.

Each computer virus infection is logged into a log 47 (shown in FIGURE 2) (block 141). If opted (block 142), a warning is generated (block 143) to the network administrator and other appropriate user. As well, if opted (block 144), a valid packet is sent as a spoof of an infected packet (block 145) by way of the outgoing packet queue 48. The routine then returns.

FIGURE 11 is a flow diagram showing the routine for correlating events 150 for use in the method 90 of FIGURE 6. The purpose of this routine is to analyze the incoming packet stream for patterns indicating a denial of service or related network attack.

Each TCP segment 75 (shown in FIGURE 5) is retrieved from the protocol-specific queues 41 (shown in FIGURE 2) (block 152). The retrieved TCP segment 75 is compared to the existing events 45 stored in the event database 44 (block 153) to detect a pattern indicative of a denial of service or related network attack. If a pattern is detected (block 154), a warning is generated (block 155). The TCP segment 75 is then enqueued back onto the appropriate protocol-specific queue 41 (block 156). Processing continues with each TCP segment 75 in the appropriate protocol-specific queue 41 (block 157), after which the routine returns.

While the invention has been particularly shown and described as referenced to the embodiments thereof, those skilled in the art will understand that the foregoing and other changes in form and detail may be made therein without departing from the spirit and scope of the invention.